13. (Amended) A data-processing system comprising:

a memory containing a proxy class, an instance of the proxy class, an interface specified at runtime having methods, and an object for handling method invocations, the proxy class implementing the interface; and

a processor for generating the proxy class at runtime, receiving a request to access a method of the interface and dispatching the request to the object to facilitate processing of the requested method of the interface.

REMARKS

In the Office Action dated July 17, 2002, the Examiner rejected claims 1-4, 6-9, and 12 under 35 U.S.C. § 102(b) as being unpatentable by Leach et al. (U.S. Patent No. 5,805,885); rejected claims 5 and 10 under 35 U.S.C. § 103(a) as being unpatentable over Leach et al. in view of Hailpern et al. (U.S. Patent No. 6,257,937); and rejected claims 11 and 13 under 35 U.S.C. § 103(a) as being unpatentable over Leach et al. in view of Hailpern et al. and further in view of Hughes (U.S. Patent No. 6,345,382).

By this amendment, Applicants have canceled claims 3 and 8 without prejudice or disclaimer and amended claims 1, 6, 12, and 13 . Based on these amendments and the following arguments, Applicants respectfully traverse the rejection of claims 1-13 under 35 U.S.C. § 102(b) and 35 U.S.C. § 103(a).

Regarding the rejection of claims 1-4, 6-9, and 12 under 35 U.S.C. § 102(b), Applicants note that Leach et al. is available as prior art under 35 U.S.C. § 102(e) and

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

not 35 U.S.C. § 102(b). Accordingly, Applicants will assume the rejection of claims 1-4, 6-9, and 12 is based on 35 U.S.C. § 102(e). Applicants respectfully request clarification or verification of the rejection of these claims in the next communication by the Examiner.

Applicants traverse the rejection of claims 1-4, 6-9, and 12 under 35 U.S.C. § 102 because Leach et al. does not teach each and every step and/or element of these claims.

Leach et al. discloses a method and system for aggregating objects within an enclosed object. The method and system allows for static and dynamic aggregation. In static aggregation, an interface of an enclosing object knows in advance (e.g., before runtime) how to return an identifier to an external interface of an enclosed object. In dynamic aggregation, an enclosed object is added to the enclosing object after the enclosing object is instantiated. Once included in the enclosing object, a reference to an interface of the enclosed object may be returned in response to an external request to access the interface (see Leach et al., Abstract, col. 11, lines 23-34 and col. 21, line 65 to col. 22, line 21).

In contrast, claim 1 recites a combination of steps including, among other things, dispatching a request to an object to facilitate processing of a method of an interface specified at runtime, and returning a result of the processed method by the object. The block diagram illustrated in Fig. 7C of Leach et al. (referenced by the Examiner) does not show the dispatching of a request to an object to facilitate processing of the method of an interface. Instead, this block diagram shows a multitype object ("MTO") after adding an interface using particular method (e.g., AddObject). The IUnknown interface

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

included in the MTO is capable of providing a pointer to an external interface and/or other MTO interfaces (see Leach et al., col. 24, lines 27-44). Further, Leach et al. does not teach returning a result of a processed method, as recited in claim 1. The only information provided by an enclosing object taught by Leach et al. is a reference or a pointer to an interface. Accordingly, because Leach et al. fails to teach every recitation of claim 1, Applicants respectfully request that the rejection of this claim under 35 U.S.C. § 102 be withdrawn and the claim allowed.

Claims 2 and 4 depend from claim 1. As explained, claim 1 is distinguishable from Leach et al. Accordingly, claims 2 and 4 are also distinguishable from this reference for at least the same reason set forth for claim 1 and Applicants request that the rejection of these claims under 35 U.S.C. § 102 be withdrawn and the claims allowed.

Further, Leach et al. does not teach specifying an object to process method invocations on the instance of the class generated at runtime, as recited in claim 4. Instead, Leach et al. discloses a system and method that provides references or pointers to interfaces of enclosed objects. Further, Leach et al. discloses providing a method for modifying a determination of which interfaces to retrieve and how to invoke them "in combination if more than one instance of the same interface is present in the aggregate object" (i.e., enclosing object) (see Leach et al., col. 23, lines 9-12). This process taught by Leach et al. is not equivalent to specifying an object to process method invocations on the instance, as recited in claim 4 because, at the very least, there is no specification of an object that processes method invocations. Moreover, the method provided by Leach et al. is limited to information on how to invoke interfaces in

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

combination if more than one instance of the same interface is present. This limitation has no relationship to the recitations of claim 4. Accordingly, Applicants respectfully request that the rejection of this claim under 35 U.S.C. § 102 be withdrawn and the claim allowed.

Claims 6 and 12 includes recitations similar to those of claim 1. As explained, claim 1 is distinguishable from Leach et al. Accordingly, claims 6 and 12 are also distinguishable from this reference for at least the same reason set forth for claim 1 and Applicants request that the rejection of these claims under 35 U.S.C. § 102 be withdrawn and the claims allowed.

Claims 7 and 9 depend from claim 6. As explained, claim 6 is distinguishable from Leach et al. Accordingly, claims 7 and 9 are also distinguishable from this reference for at least the same reason set forth for claim 1. Further, claim 9 includes recitations similar to those of claim 4. Accordingly, claim 9 is also distinguishable from Leach et al. for at least the same reasons set forth for claim 4. Based on the foregoing, Applicants request that the rejection of claims 7 and 9 under 35 U.S.C. § 102 be withdrawn and the claims allowed.

Applicants traverse the rejection of claims 5 and 10 under 35 U.S.C. § 103(a) because a prima facie case of obviousness has not been made by the Examiner. To establish a prima facie case of obviousness under 35 U.S.C. § 103(a), each of three requirements must be met. First, the reference or references, taken alone or combined, must teach or suggest each and every element recited in the claims (See M.P.E.P. § 2143.03 (8th ed. 2001).) Second, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

skill in the art to combine the referenced in a manner resulting in the claimed invention. Third, a reasonable expectation of success must exist. Moreover, each of these requirements must "be found in the prior art, and not be based on applicant's disclosure" (See M.P.E.P. § 2143 (8th ed. 2001)).

As explained above, Leach et al. discloses a method and system for aggregating objects within an enclosed object. The Examiner admits that Leach et al. does not teach an invocation handler. To supplement this missing recitation, the Examiner asserts that the process execution handler taught by Hailpern et al. is equivalent to an invocation handler and that it would have been obvious to apply the execution handler to the system of Leach et al." because this allows dynamic aggregating objects" (see Office Action, page 4, lines 7-10). Applicants respectfully disagree with the Examiner.

The process execution handler taught by Hailpern et al. carries out "necessary processing" for a server that performs virus checking in a server network environment (see Hailpern et al., col. 15, lines 48-40). The execution handler is not an invocation handler that receives method invocations, as recited in claims 5 and 10.

Further, neither Hailpern et al. or Leach et al., alone or in combination, teach or suggest generating at runtime a class that implements the interface by generating code for each of the methods included in the interface, as recited in claims 5 and 10. The code table 3 taught by Leach et al., referred to by the Examiner, is a listing of pseudo code for a class definition on an object enclosed in an aggregate object (see Leach et al., col. 14, lines 23-28). There is no mention or suggestion in Leach et al., particularly in code table 3, of generating a class at runtime that implements an interface by generating code for each of the methods in the interface, as recited in claims 5 and 10.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

Accordingly, because Hailpern et al. or Leach et al., alone or in combination, fail to teach or suggest the recitations of claims 5 and 10, Applicants request that the rejection of these claims under 35 U.S.C. § 103(a) be withdrawn and the claims allowed.

Additionally, there is no suggestion or motivation in the references themselves or in the knowledge generally available to one of ordinary skill in the art to combine Hailpern et al. and Leach et al., in a manner resulting in the recitations of claims 5 and 10. According to the Examiner's position, the process execution handler taught by Hailpern et al. would "allow dynamic aggregating objects at runtime" (see Office Action, page 4, lines 9-10). Applicants disagree for the following reasons. First, there is no evidence that the process execution handler taught by Hailpern et al. performs any functions other than virus checking processes identified by a target determining handler. Accordingly, Hailpern et al. does not teach a system or method that even suggest aggregating objects, much less doing so dynamically. Second, based on the Examiner's position, combining the teachings of Hailpern et al. with Leach et al. appears to provide no additional capabilities, advantages, or benefit to the system and method taught by Leach et al. According to the Examiner, combining these references "allows dynamic aggregating objects at runtime." Leach et al. discloses a system and method that already performs dynamic aggregating of objects. Accordingly, one skilled in the art would not have been motivated to combine the teachings of Hailpern et al. with Leach et al. because there would be no reason for the combination.

Based on the foregoing, Applicants request that the rejection of claims 5 and 10 under 35 U.S.C. § 103(a) be withdrawn and the claims allowed.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

Claims 11 and 13 include recitations similar to those of claims 1 and 5. As explained, claims 1 and 5 are distinguishable from Leach et al. and Hailpern et al. Accordingly, claims 11 and 13 are also distinguishable from these references for at least the same reason set forth for claims 1 and 5. Further, claims 11 is also distinguishable from Hughes because that reference does not teach or suggest, alone or in combination with Leach et al. and Hailpern et al., at least dispatching a request to an invocation handler object and returning a value from the handler object to a proxy class instance, as recited in the claim 11. Further, these references fail to teach a proxy class generated at runtime as recited in claims 11 and 13. The proxy class disclosed by Hughes is not generated at runtime. Instead, a manufacturer generates code for interfaces, and other functionalities associated with the proxy class (see Hughes, col. 4, lines 18-31 and 60-67. As disclosed by Hughes, a proxy class is not generated at runtime, but rather a user dynamically specifies at runtime the "customization" of an instance of the proxy class (see Hughes, col. 8, lines 6-8). Accordingly, Hughes, Leach et al. and Hailpern et al., alone or in combination, fail to teach or suggest the recitations of claims 11 and 13 and Applicants request that the rejection of these claims under 35 U.S.C. § 103(a) be withdrawn and the claims allowed.

In view of the foregoing amendments and remarks, Applicants respectfully request the reconsideration and reexamination of this application and the timely allowance of claims 1, 2, 4-7, and 9-13.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

Please grant any extensions of time required to enter this response and charge any additional required fees to our deposit account 06-0916.

Respectfully submitted,

Dated: October 7, 2002

By: _____
Joseph E. Palys
Reg. No. 46,508

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

APPENDIX TO CLAIM AMENDMENTS

1. (Amended) A method in a data processing system comprising the steps of:

generating at runtime a class that implements an interface specified at runtime having a method;

creating an instance of the class;

receiving by the class instance a request to process the method of the interface; [and]

dispatching the request to an object to facilitate processing of the method of the interface; and

returning a result of the processed method by the object.

6. (Amended) A computer-readable medium containing instructions for controlling a data processing system to perform a method comprising the steps of:

generating at runtime a class that implements an interface specified at runtime having a method;

creating an instance of the class;

receiving by the class instance a request to process the method of the interface; [and]

dispatching the request to an object to facilitate processing of the method of the interface; and

returning a result of the processed method by the object.

12. (Amended) A data-processing system comprising:

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

means for generating at runtime a class that implements an interface specified at runtime having a method;

means for creating an instance of the class;

means for receiving by the class instance a request to process the method of the interface; [and]

means for dispatching the request to an object to facilitate processing of the method of the interface; and

means for returning a result of the processed method by the object.

13. (Amended) [The] A data-processing system comprising:

a memory containing a proxy class, an instance of the proxy class, an interface specified at runtime having methods, and an object for handling method invocations, the proxy class implementing the interface; and

a processor for generating the proxy class at runtime, receiving a request to access a method of the interface and dispatching the request to the object to facilitate processing of the requested method of the interface.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com